Q1 Identifying Sorts

3 Points

Q1.1

1 Point

Below you will find intermediate steps in performing a sorting algorithms on the given input array. The steps do not necessarily represent consecutive steps in the algorithm (that is, many steps are missing), but they are in the correct sequence. Select the algorithm it illustrates from among the following choices.

Input Array

```
1429, 3291, 7683, 1337, 192, 594, 4242, 9001, 4392, 129, 1000
```

During execution

```
1429, 3291, 7683, 192, 1337, 594, 4242, 9001, 4392, 129, 1000
```

```
1429, 3291, 192, 1337, 7683, 594, 4242, 9001, 129, 1000, 4392
```

```
192, 1337, 1429, 3291, 7683, 129, 594, 1000, 4242, 4392, 9001
```

- In-Place Heapsort
- Selection Sort
- Merge Sort
- Insertion Sort

Save Answer

Q1.2
1 Point

Suppose we're executing an unknown sorting algorithm on an array of 8 integers. Consider the following series of steps that represent the order of elements at certain points in time.

Input Array

`0, 4, 2, 7, 6, 1, 3, 5`

During execution

`0, 2, 4, 7, 6, 1, 3, 5`

Sorted result

`0, 1, 2, 3, 4, 5, 6, 7`

- In-Place Heapsort
- Selection Sort
- Merge Sort
- Insertion Sort

Save Answer

Q1.3
1 Point

Suppose we're executing an unknown sorting algorithm on an array of 8 integers. Consider the following series of steps that represent the order of elements at certain points in time.

Input Array

`0, 4, 2, 7, 6, 1, 3, 5`

During execution

`0, 1, 2, 3, 6, 4, 7, 5`

Sorted result

`0, 1, 2, 3, 4, 5, 6, 7`

- In-Place Heapsort
- Selection Sort
- Merge Sort
- Insertion Sort

Save Answer

Q2 Time Complexity in Special Cases

5 Points

Suppose we run a sorting algorithm on an array of integers that every number is the same, for example, {1,1,1,1}.
Select the ***tight*** asymptotic running time for each of the following sorting algorithms:

Q2.1 Insertion Sort

1 Point

$O(1)$

$O(logN)$

$O(N)$

$O(NlogN)$

$O(N^2)$

Save Answer

Q2.2 Selection Sort

1 Point

$O(1)$

$O(logN)$

$O(N)$

$O(NlogN)$

$O(N^2)$

Save Answer

Q2.3 Heap Sort
1 Point

$O(1)$

$O(logN)$

$O(N)$

$O(NlogN)$

$O(N^2)$

Save Answer

Q2.4 Merge Sort
1 Point

$O(1)$

$O(logN)$

$O(N)$

$O(NlogN)$

$O(N^2)$

Save Answer

Q2.5 Quick Sort
1 Point

$O(1)$

$O(logN)$

$O(N)$

$O(NlogN)$

$O(N^2)$

Save Answer

Q3 Stable Sorts
4 Points

Consider the following Java class:

```
public class PlayingCard {
    public String suit;
    public int rank;
    public PlayingCard(String suit, int rank) {
        this.suit = suit;
        this.rank = rank;
    }
    public int compareTo(PlayingCard other) {
        return suit.compareTo(other.suit);
    }
}
```

This class is intended to represent an arbitrary French Playing Card, so we restrict suit to be one of ”spades”, ”clubs”, ”hearts”, or ”diamonds”, and restrict rank to a value between 1 and 13 (inclusive). Beyond these restrictions, you do not have to be familiar with French playing cards to solve this problem.

Note: this problem makes no assumptions about how these cards are being used – in particular, do not assume that all `PlayingCard` objects in the problem need to be from the same deck of cards.

In this problem, assume that `PlayingCard` objects are indistinguishable from one another if the values of their fields are equal. (That is, assume that we will never use == or care about any object references themselves when we examine the difference between sorting algorithm results).

Q3.1 Same output
2 Points

Suppose we want to run a sorting algorithm on a list of 5 PlayingCard objects that uses the `compareTo` method to compare any two cards (note that it simply uses the underlying `compareTo` of the `String` suit field).

Select all the answers below that provide a group of 5 `PlayingCard` objects where, if they were placed in a list in any order and fed into a sorting algorithm, Insertion sort and Selection sort would always result in the same output.

(Hint: recall that Insertion sort is a stable sort, but Selection sort is not.)

- [ ] PlayingCard("spades", 1), PlayingCard("spades", 2), PlayingCard("spades", 3), PlayingCard("spades", 4), PlayingCard("spades", 5)
- [ ] PlayingCard("spades", 1), PlayingCard("clubs", 3), PlayingCard("diamonds", 4), PlayingCard("hearts", 8), PlayingCard("diamonds", 9)
- [ ] PlayingCard(”hearts”, 1), PlayingCard(”hearts”, 1), PlayingCard(”spades”, 1), PlayingCard(”clubs”, 1), PlayingCard(”diamonds”, 1))
- [ ] PlayingCard("clubs", 1), PlayingCard("clubs", 1), PlayingCard("diamonds", 1), PlayingCard("hearts", 1), PlayingCard("spades", 1)
- [ ] PlayingCard("clubs", 1), PlayingCard("diamonds", 1), PlayingCard("hearts", 1), PlayingCard("spades", 1), PlayingCard("spades", 2)

Save Answer

Q3.2 Different outputs
2 Points

Now, select all the answers below that provide an ordered list of 5 `PlayingCard` objects where, if they were fed into a sorting algorithm, Insertion sort and Selection sort would result different outputs.

- [ ] PlayingCard(”diamond”, 1), PlayingCard(”diamond”, 2), PlayingCard(”clubs”, 1), PlayingCard(”spades”, 1), PlayingCard(”spades”, 3)
- [ ] PlayingCard("diamonds", 1), PlayingCard("spades", 1), PlayingCard("diamonds", 1), PlayingCard("hearts", 1), PlayingCard("clubs", 1)
- [ ] PlayingCard("spades", 5), PlayingCard("diamonds", 3), PlayingCard("diamonds", 2), PlayingCard("heart", 4), PlayingCard("clubs", 1)
- [ ] PlayingCard("spades", 1), PlayingCard("clubs", 1), PlayingCard("diamonds", 1), PlayingCard("hearts", 1), PlayingCard("clubs", 2)
- [ ] PlayingCard("diamonds", 2), PlayingCard("diamonds", 1), PlayingCard("spades", 1), PlayingCard("clubs", 1), PlayingCard("hearts", 1)

Save Answer

Q4 Sorting Design Decisions

9 Points

For each of the following scenarios, choose the best sorting algorithm(s) from the list below. Then, justify your choice by describing what properties of that sorting algorithm make it the best choice, and why those properties are useful in this particular scenario.

Maximum Length Per Question: 2-3 sentences.

Sorting algorithms: Insertion sort, Selection sort, Heap sort, Merge sort, Quick Sort

Q4.1 Game Programming

3 Points

You are a game programmer in the 1980s who is working on displaying a sorted list of enemy names that a player has encountered during their gameplay. Since it is a game, you want to display the names of the enemies as fast as possible, but because it is the 1980s, your customers are used to and will be okay with occasional slow loading times. Additionally, the game is intended to run normally on consoles that don't have much memory available.

Save Answer

Q4.2 Computer Files
3 Points

Imagine that you are sorting a small set of computer files by their file name. You realize, however, that each computer file is huge and takes up a lot of disk space, so you do not want to copy excessively when sorting. In fact, even just moving and rearranging these large files is expensive, so you don't want to move them often.

(Hint: You may find it useful to refer to visuals of the sorting algorithms. Lecture slides and https://visualgo.net/en/sorting are good resources for remembering these general ideas.)

Save Answer

Q4.3 NASA Probe
3 Points

Imagine that you are a NASA software engineer. You're assigned a task to sort data you receive from a probe on Mars, in which each piece of data includes time and temperature. The sensors on this probe capture very large amounts of data. The data is already given to you in sorted order of earliest to latest time, but you want to sort them by temperature, where ties in temperature are ordered by time.

Save Answer

Save All Answers

Submit & View Submission >